
# NASA Goddard Software Engineering Workshop

## NASA Software Tools for High-Quality Requirements Engineering

**James R. McCoy**
SRS Information Services
NASA Software Assurance Technology Center
http://satc.gsfc.nasa.gov
james.mccoy@gsfc.nasa.gov

# Overview

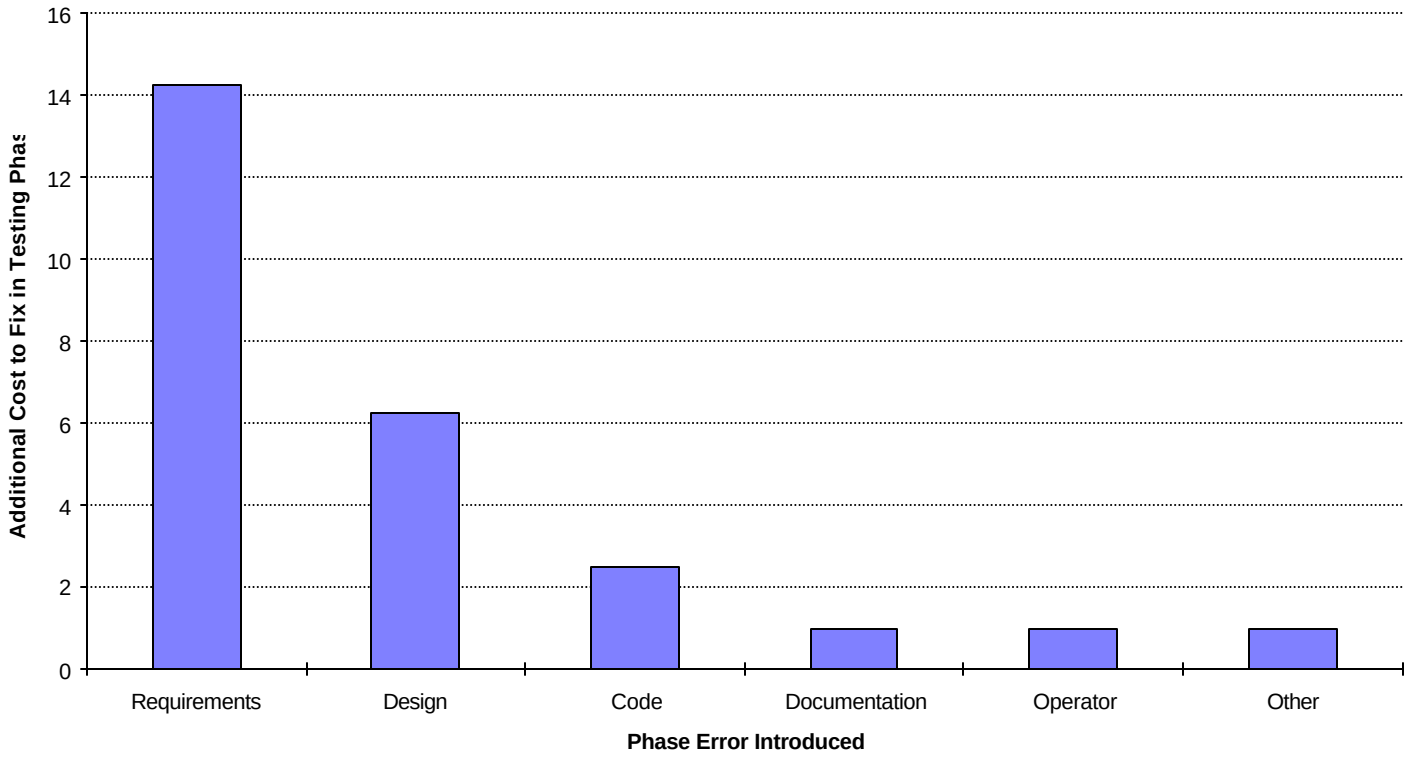
- ARM – Automated Requirements Measurement Tool
- SCAT – Safety Critical Analysis Tool
- RUT – Requirements Use case Tool

# Automated Requirements Measurement Tool

*an early life cycle tool for assessing requirements that are specified in natural language*

# The Price of Errors

## Cost to Fix Errors Found in Testing Phase




**\ Find errors as soon as possible for maximum savings !**

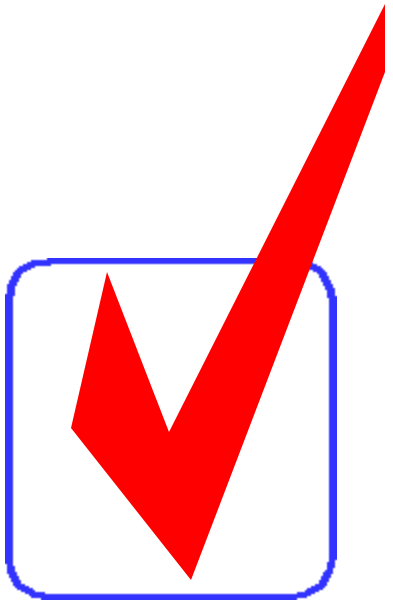
# Problems Common to Most Documents

- Specification
  - Documentation and style standards not used or misapplied
  - Poor organization of information content
  - Uneven emphasis and levels of detail
  - Inconsistent identification schemes
- Statements
  - Verbose text
  - Poor sentence structure
  - Poor word selection
  - Diagrams, tables, charts, unclear

# SRS Quality Attributes

An SRS should be:

- Complete
- Consistent
- Correct
- Modifiable
- Concise
- Testable
- Traceable
- Unambiguous
- Understandable
- Validatable
- Verifiable
- Independent
- Annotated
- at the Appropriate Level of Abstraction



# Imperatives

- **Imperatives** are those words and phrases that command that something must be provided.
  - ***Shall*** is usually used to dictate the provision of a functional capability.
  - ***Must*/*must not*** is used to establish performance requirements or constraints.
  - ***Are applicable*** is used to include, by reference, standards or other documentation as an addition to the requirements being specified.
  - ***Responsible for*** is used in requirements documents that are written for systems whose architectures are predefined.
  - ***Will*** is used to cite things that the operational or development environment are to provide to the capability being specified.
  - ***Is required to*** is passive voice; ***Should*** is advisory. Neither should be used in requirement specification statements.

# Weak Phrases

- **Weak Phrases** are clauses that are apt to cause uncertainty and leave room for multiple interpretations.
  - Phrases such as *adequate*, *as appropriate* and *timely* indicate that what is required is either defined elsewhere or, worse, that the requirement is open to subjective interpretation.
  - Phrases such as *but not limited to*, *as a minimum*, and *TBD* provide a basis for expanding a requirement or adding future requirements.

# Options

- **Options** (such words as *may* and *optionally*) give the developer latitude in satisfying the specification statements that contain them.
  - Options loosen the specification,
  - Reduces the acquirer's control over the final product, and
  - Establishes a basis for possible cost and schedule risks.

# Generalities

- **Generalities** provide gross quantitative or qualitative descriptors that indicate direction of intent but no useful information.

  - *About*
  - *Adequate*
  - *Almost*
  - *At a minimum*
  - *Bad*
  - *Close*
  - *Good*
  - *Many*
  - *Most*
  - *TDB*
  - *Timely*

# Requirement Attributes & Metrics

- **Ambiguity** = Weak Phrases (adequate, as appropriate, as applicable, but not limited to, normal, if practical, timely, as a minimum) + Options (can, may, optionally)
- **Completeness** = TBD + TBA + TBS + TBR
- **Understandability** = Numbering Scheme
- **Traceability** = Number of Items traced to tests, between builds, between levels of detail
- **Volatility** = Number of Changes / Number of Requirements
- **Number of Requirements** = Imperatives (shall, must, will, required, responsible for, should, are to, are applicable) + Continuances (below:, as follows:, following:, listed:, in particular, support:, : )

Software Assurance Technology Center

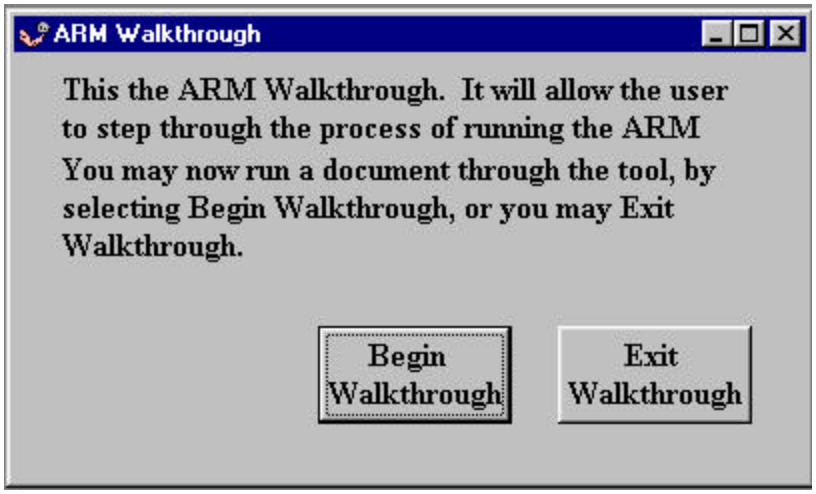
# ARM Analysis

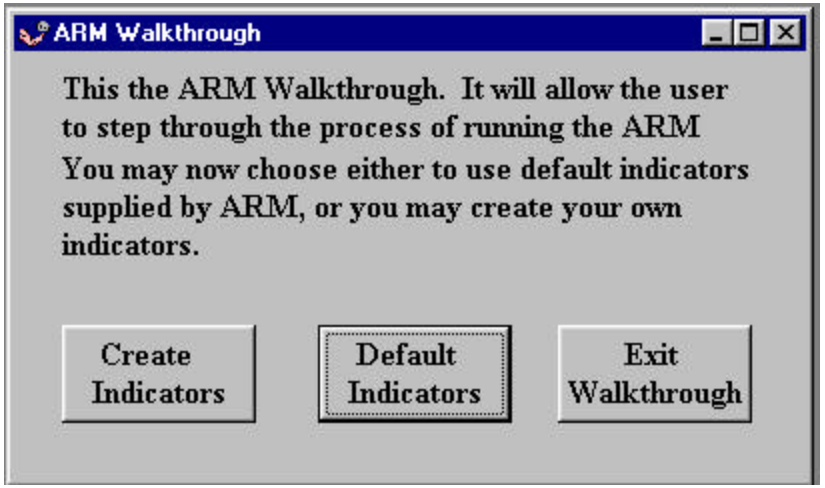
| 56 DOCUMENT | **Lines of Text** - Count of the physical lines of text | **Imperatives** - shall, must, will, should, is required to, are applicable, responsible for | **Continuances** - as follows, following, listed, inparticular, support | **Directives** - figure, table, for example, note: | **Weak Phrases** - adequate, as applicable, as appropriate, as a minimum, be able to, be capable, easy, effective, not limited to, if practical | **Incomplete** (TBD, TBS) | **Options** - can, may, optionally |
|---|---|---|---|---|---|---|---|
| **Minimum** | 143 | 25 | 15 | 0 | 0 | 0 | 0 |
| **Median** | 2,265 | 382 | 183 | 21 | 37 | 7 | 27 |
| **Average** | 4,772 | 682 | 423 | 49 | 70 | 25 | 63 |
| **Maximum** | 28,459 | 3,896 | 118 | 224 | 4 | 32 | 130 |
| **Stdev** | 759 | 156 | 99 | 12 | 21 | 20 | 39 |
| | | | | | | | |
| **Project X** | 34,664 | 1,176 | 714 | 873 | 13 | 480 | 187 |

# Requirements Summary

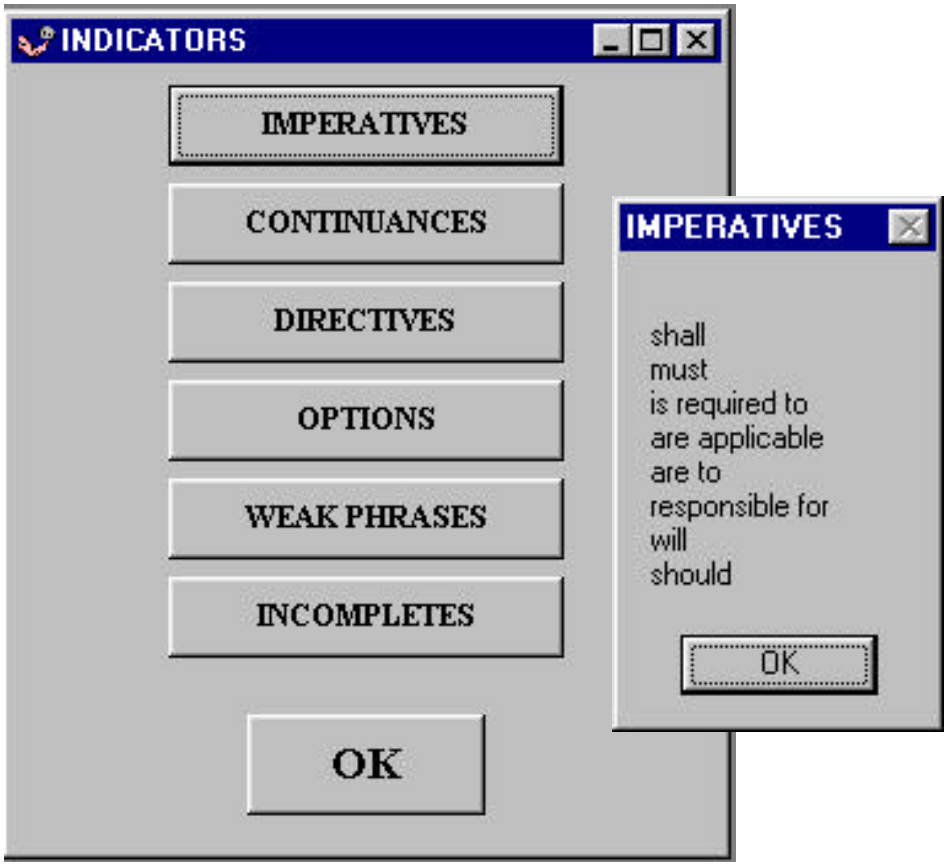
- Use of natural language for requirements may result in problems later; need care, attention, and review of language usage and structure.
- Metrics can be used to track requirements process and give valuable insight into project status and early warning of problems.

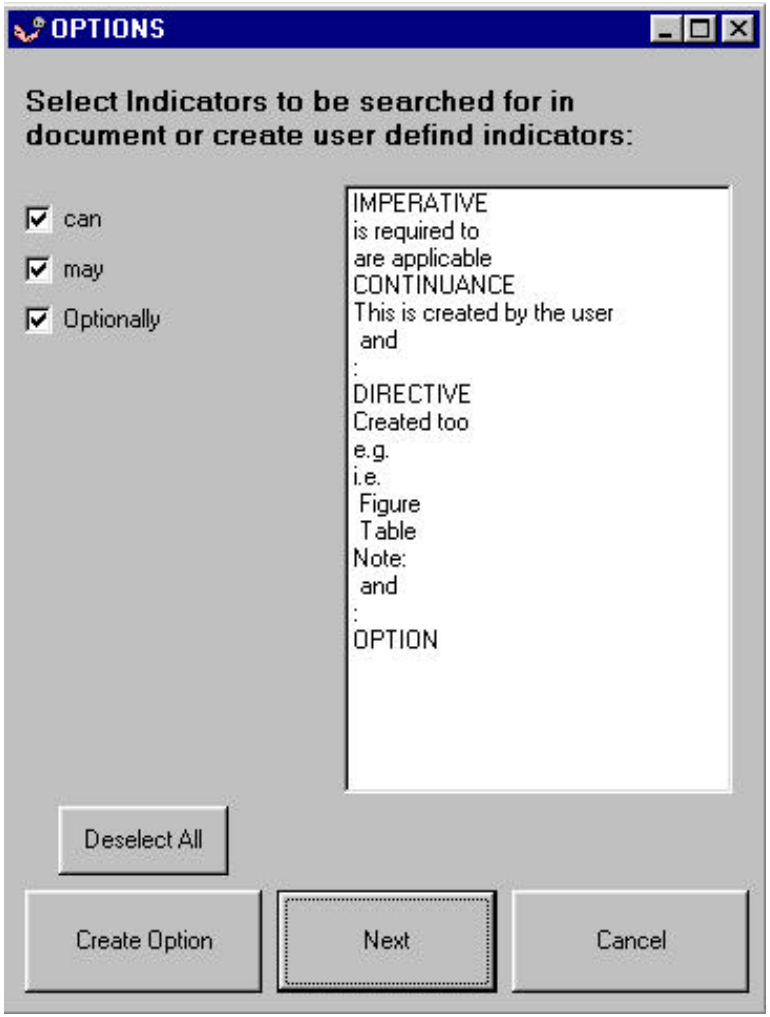
# ARM Screen Shots

# ARM Screen Shots



# Safety Critical Analysis Tool

*scans documents and evaluates them lexically by searching for indicators that relate potential hazards to their controlling systems, subsystems, and components*

Software Assurance Technology Center

# SCAT Overview

- SCAT uses specified indicators to identify systems, subsystems, and components that potentially can impact safety.
- This information enables responsible entities at NASA HQ as well as the various NASA centers to monitor identified systems more effectively, thus, assuring that safety-related requirements have visibility and are adequately addressed.

Software Assurance Technology Center

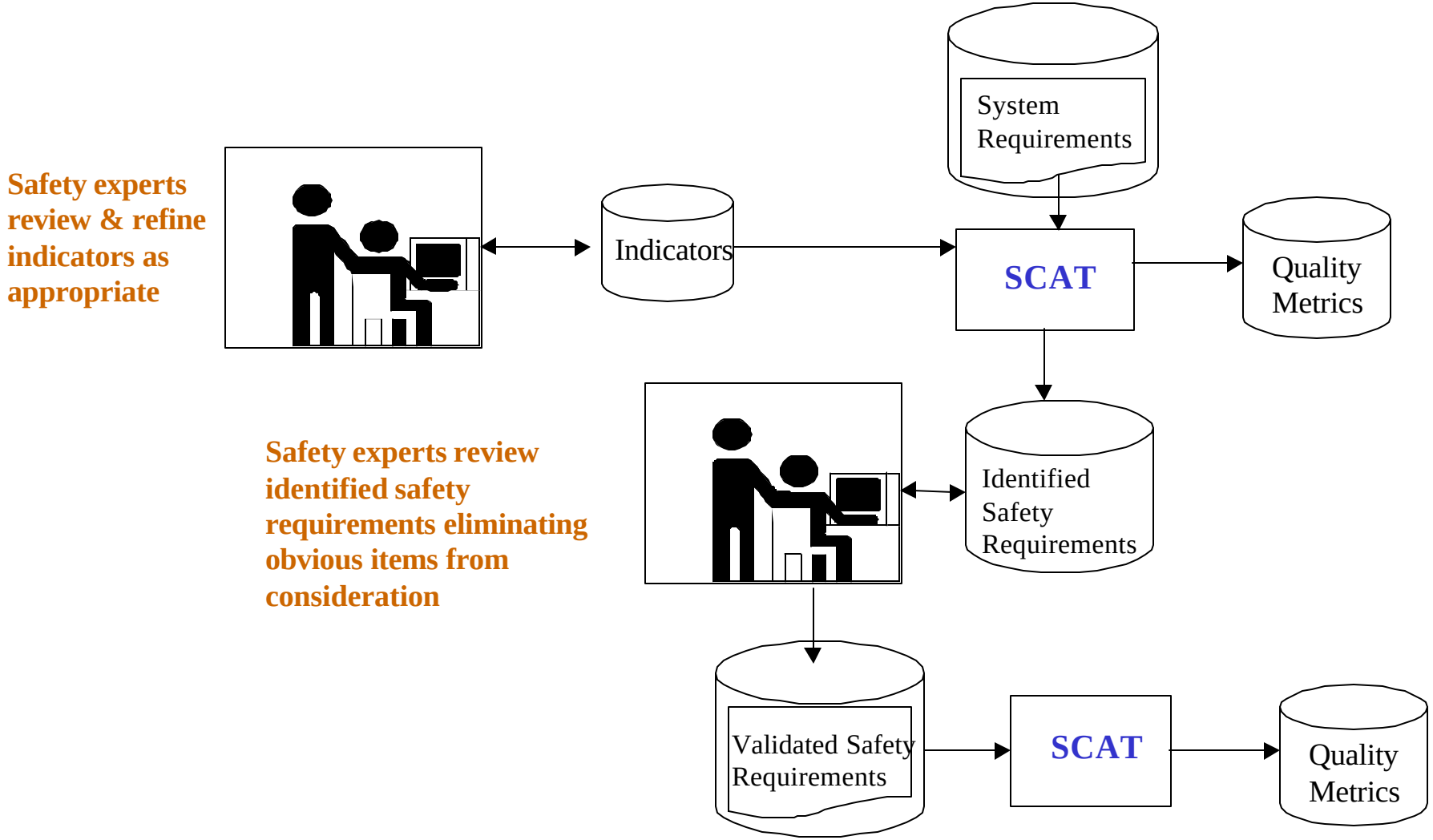
# SCAT Overview

- SCAT is intended to be applicable to the entire system life cycle; e.g., research, technology development, design, test and evaluation, production, construction, checkout/calibration, operation, maintenance and support, modification, and disposal.
- Current work builds on experience gained in development and use of the ARM tool.
- This technology will be applied analogously to project documents in identifying potential system safety requirements.

# Sample Indicators of Safety Requirements

- Availability
- Burn
- Combustion
- Decontaminate
- Explosion
- Failure
- Hazardous
- Inadvertent activation
- Line rupture
- Must work
- Out-of-tolerance
- Personnel exposure
- Quality assurance
- Radiation
- Spontaneous ignition
- Toxic
- Unsafe
- Vulnerable
- Warning

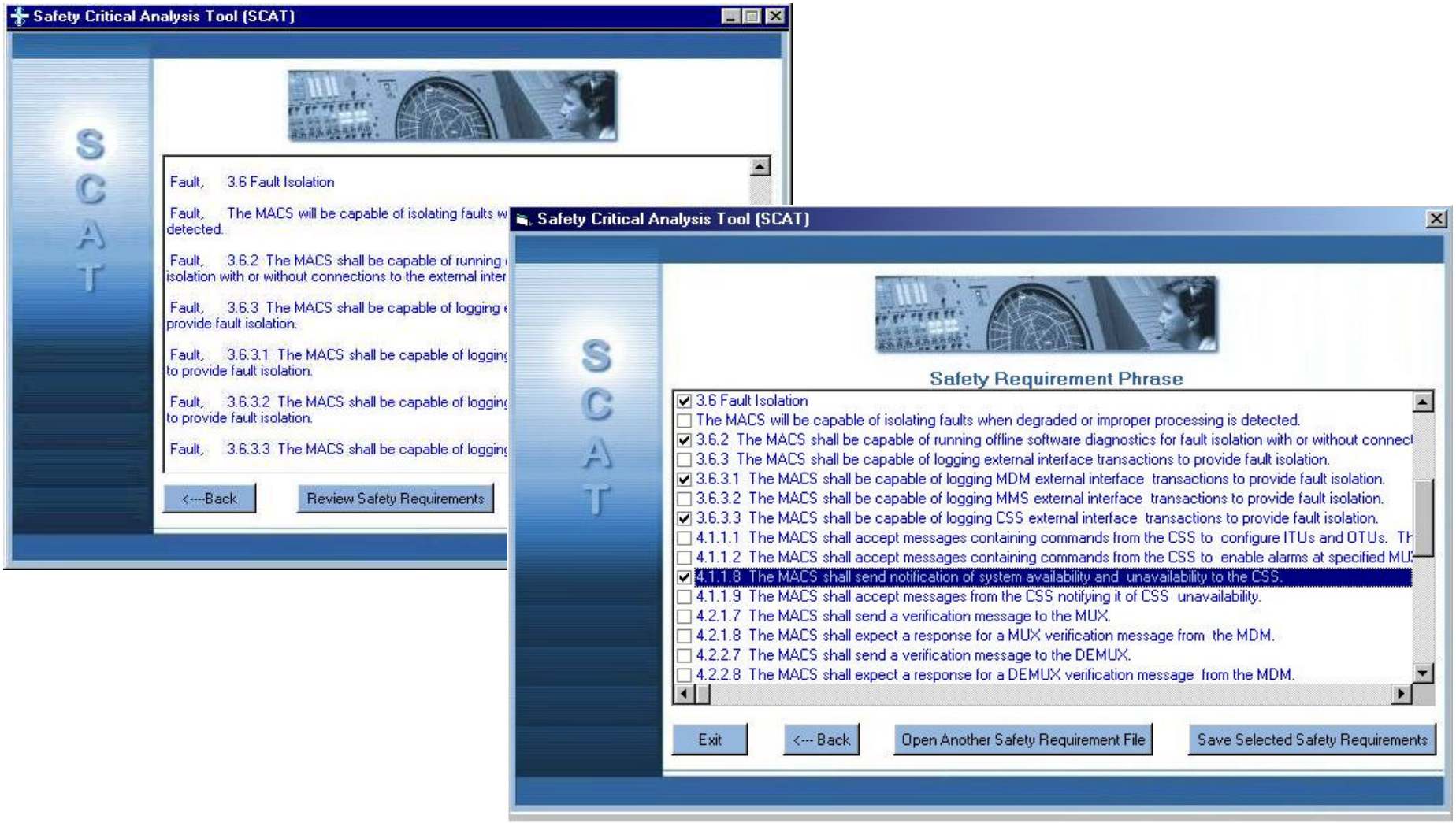
# SCAT Operational Procedure



# SCAT Screen Shots

# SCAT Screen Shots

# SCAT Screen Shots



# RUT

## Requirements Use case Tool

*provides a standard template and repository for the specification and storage of text-based use case requirements*

# Use Case Overview

- System requirements are the foundation upon which an entire system is built.
- Traditional vehicle for capturing and communicating requirements is the Software Requirements Specification (SRS).
- Use cases provide a more user-centered approach for specifying requirements.

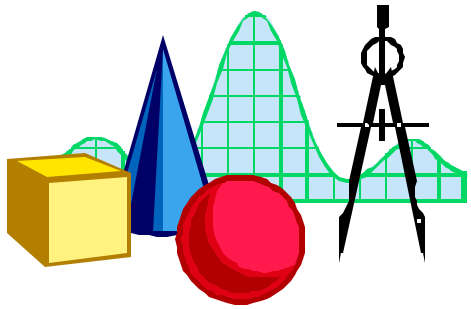
# Unified Modeling Language

- The Unified Modeling Language (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of complex software systems.

- The UML:
  - Is a *language*.
  - Applies to *modeling* and systems.
  - Is based on the object-oriented paradigm.

# Use Case Model

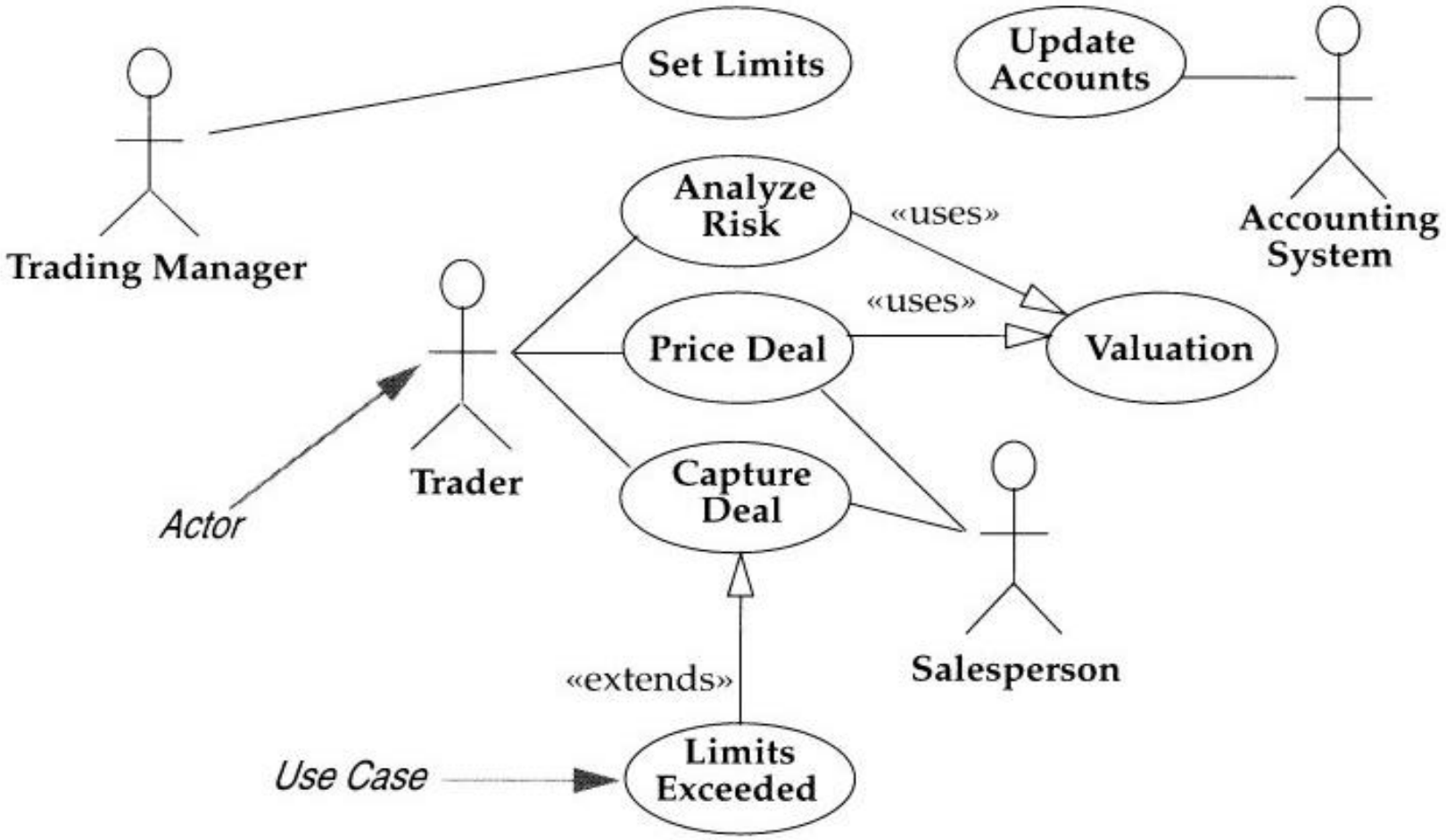
- Use cases were designed to capture, via a combination of structured text and graphics, the functional requirements of a system.
- Use cases are usually described in a textual document that accompanies a use case diagram. The combination of these use case diagrams and their supporting documentation is known as a *use case model*.



# Use Case Model

- Use case models:
  - Illustrate a system's intended functions (use cases), its surroundings (actors), and the relationships between them (use case diagrams).
  - Are used to COMMUNICATE.
  - Provide a vehicle used by customers and developers to discuss the system's functionality.

# UML Use Case Diagram



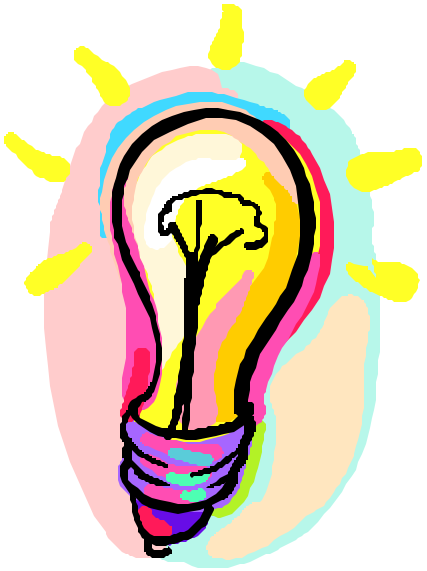

Fowler, M., & Scott, K., UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, 1997.

# Research Problem

- Research has been conducted on writing effective software requirements in a natural language and has resulted in the development of a tool for evaluating them.
- Use cases provide a more methodological basis for specifying and managing understandable, buildable, and verifiable functional requirements, but there is no clear evaluation technique for requirements written as use cases.

# Solution

- Identify the attributes of a quality use case.
- Develop software tool for analyzing use cases based on these characteristics.



# Quality Use Cases

- Use cases are written as natural language text descriptions expressed informally. The descriptions express *what* happens from the user's point of view. The details of *how* the system works internally are irrelevant to a use case.
- It is preferable to have actions numbered and starting on new lines. This keeps the narrative clear, improves traceability from requirements to design or test, and allows specific line references needed in the Extensions section.

# Validating Use Cases

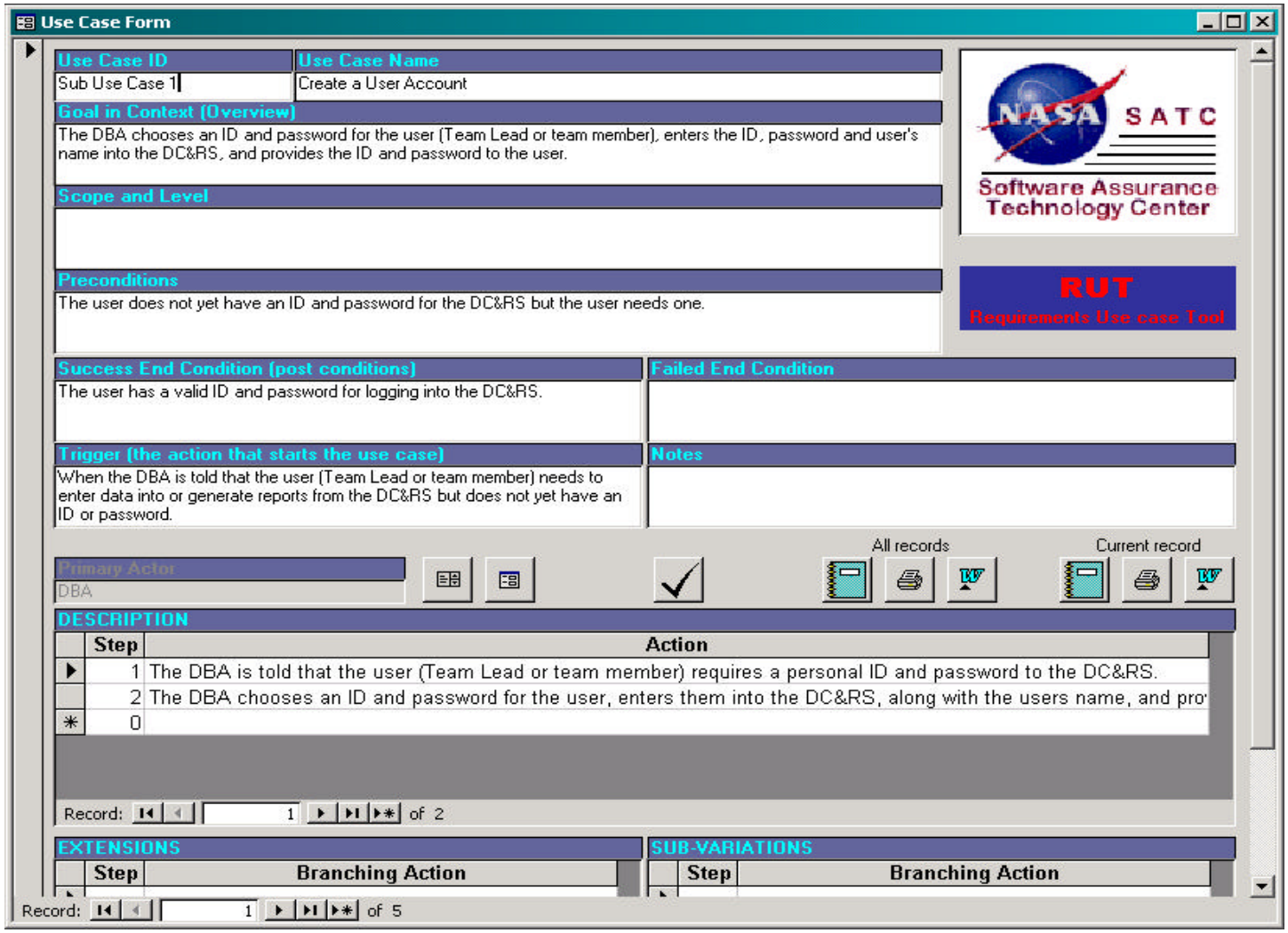
- Is the use case complete?
- Is the actor's goal going to be met?
- Are there any changes that would simplify the process depicted in the use case?
- Are there any additional goals that are not addressed?
- Are there any additional actors that are not represented?

Software Assurance Technology Center

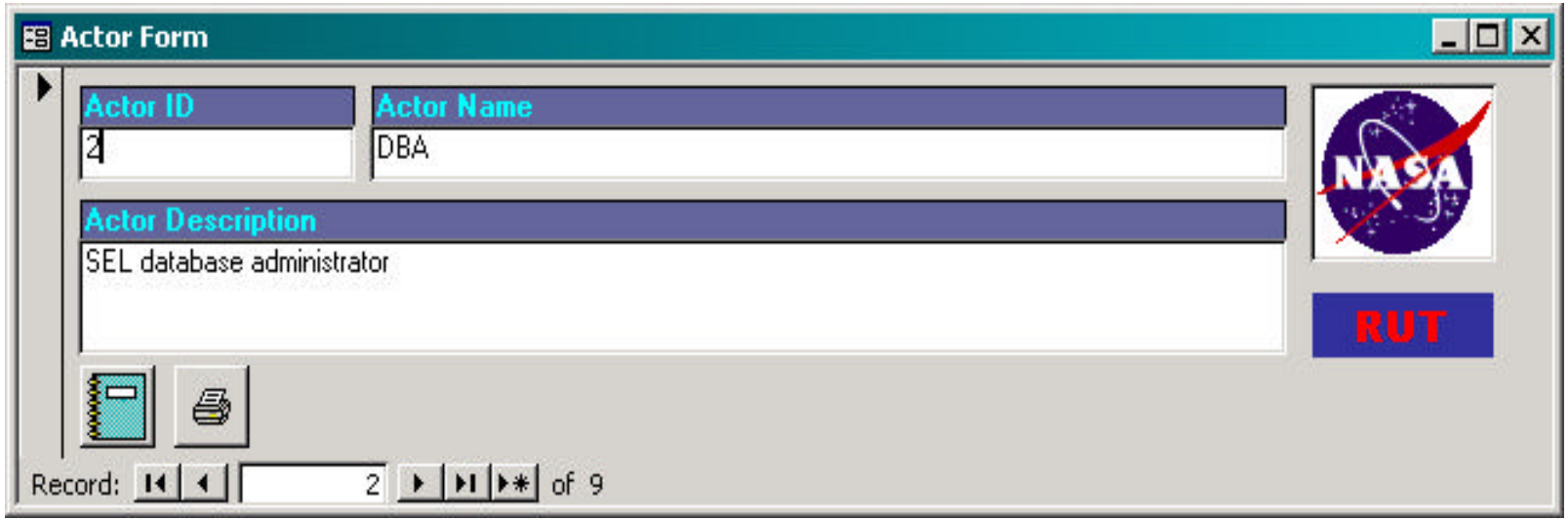
# RUT Features

- Use case repository.
- Standard use case template.
- Integration with Rational Rose.
- Mapping and numbering consistency.
- Pop-up validation questions.

# RUT Screen Shot



# RUT Screen Shot



NASA SATC
Software Assurance Technology Center

# Summary

- Extensive research conducted on writing quality requirements in a natural language resulted in the **ARM** tool for evaluating them.
- Concepts of ARM were applied to the area of systems safety with the **SCAT** tool.
- Use cases provide a more methodological basis for specifying quality requirements; a use case template and repository are provided by the **RUT** tool.

# For More Information...

- Visit Our Website:
  - http://satc.gsfc.nasa.gov/tools/.
- Or Contact:
  - ARM: Michele Crispell, michele.crispell@gsfc.nasa.gov.
  - SCAT: Peter Legowski, peter.legowski@gsfc.nasa.gov.
  - RUT: James McCoy, james.mccoy@gsfc.nasa.gov.